IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
8-10 September 2003, Lviv, Ukraine

A Tool for the Automatic Design of Electronic Control Systems and Circuits for Manufacturing Plants

A. Lamas, R. J. Duro
Grupo de Sistemas Autónomos, Universidade da Coruña, alamas@cdf.udc.es, richard@udc.es

Abstract: *In this paper we have developed a tool for automatically designing distributed digital circuits for the control of all the elements in a manufacturing plant. These circuits can be implemented as traditional boards or programmed into the controllers of the machinery present. The tool is based on evolutionary techniques and provides a way to obtain the best set of controllers for the different elements in the plant using as evaluating criteria parameters related to the global operation of the plant and not to particular parameters of the electronic circuits or individual controllers. These parameters may be productivity, cost, or any other ratio having to do with the real operation of the bussiness. In this work we have extended the evolutionary methodologies in order to be able to design, at the complete system level, combinations of low level systems (digital electronic circuits) without any direct specification of their input/output relationships but rather taking into account the plant they are going to be working in and the high level constraints imposed on the whole system.*
The resulting tool has been tested in a real kitchen furniture manufacturing plant using as test bench the lacquering line within the plant.

Keywords: - Manufacturing plant control, evolution, genetic algorithms

1. INTRODUCTION

Traditionally the design process in the realm of engineering of complex systems has involved a recursive trial and error procedure with the aid of the engineer's experience and different CAD and simulation applications. The more complex a design becomes, the more difficult it is to explore all the options available to the engineers working on it. The search space becomes huge and impossible to explore efficiently. As a consequence, the current design methodologies involve the divide and conquer strategy, that is, let us divide the design into sub-modules of reasonable complexity, design these and hope that when they are linked together the whole system works the way we wanted it to. The basic premise is that we can figure out the way of linking the modules so that they will perform the assigned task.

This approach has proven to be reasonable in most cases, but it presents two basic drawbacks. On one hand, it is based on the designer's (human) perception of how things should be. This perception is driven by experience, intuition, but not exhaustive exploration. On the other, it is almost impossible to optimize designs, as all non-linear interactions are usually ignored. In fact, as Jim Torresen indicates [1] we may very well soon see a limit in designability. In other words, in order to make the designs manageable to the human brain, they usually consist of few modules with simple interactions among them. Thus it would be desirable to take the human designer out of the loop, or, at least minimize its participation to the enunciation of the general specifications of the system in order to really be able to address complex non linear systems. Consequently, it would be necessary to provide some other type of strategy that would allow the design procedure to become completely automated.

Several authors in the literature have identified this problem and provided conceptual solutions for limited domains. In the late eighties and early nineties some authors, such as Harvey et al. [2], Cliff et al. [3] and Beer and Gallagher [4] proposed artificial evolution as a means to automate the design procedure of these types of systems, leading to the concept of Evolvable Hardware, introduced by Higuchi [5]. The basic idea of this approach is to decide on the input/output relationships in our system, provide some building blocks and let a genetic algorithm or some type of evolutionary strategy come up with a composition of these building blocks which minimizes the error for this training set. One of the fields where these concepts have been extensively applied is that of electronic circuit design. Many authors have resorted to the specification of the required electronic circuit as a set of input/output pairs and through parameters controlling FPGAs (Field Programmable Gate Arrays) or some other type of electronic structure proceeded to evolve the optimal circuit. Examples of these are the digital filter design system by Miller [6], the Adaptive Equalizer by Murakawa [7] or the methodologies by Lohn et al. [8] or the one by Torresen applied to a prosthetic hand [9]. Even though these approaches have been highly successful, they do not address the main problem of automatic design in complex real manufacturing plants and that is that, in general, in a real plant, there will coexist several electronic systems combined with other types of actuators and sensors and the input/output pairs, that is, the information one has on the plant will be for the whole combination of systems. In fact, these input/output pairs will make no reference to the variables one would like to handle at the electronic circuit level (voltages, currents, activations, etc…) but rather they will refer to ambiguous terms such as productivity, costs, breakdowns, pile ups, etc…The real plant designer works, at the plant level, with concepts like increasing productivity while minimizing breakdowns and avoiding pile ups at the inputs of certain machines, taking into account that cost is an additional variable.

In this work we have extended the evolutionary methodologies in order to be able to design, at the complete system level, combinations of low level systems (digital electronic circuits) without any direct specification of their input/output relationships but rather taking into account the plant they are going to be working in and the high level constraints imposed on the whole system.

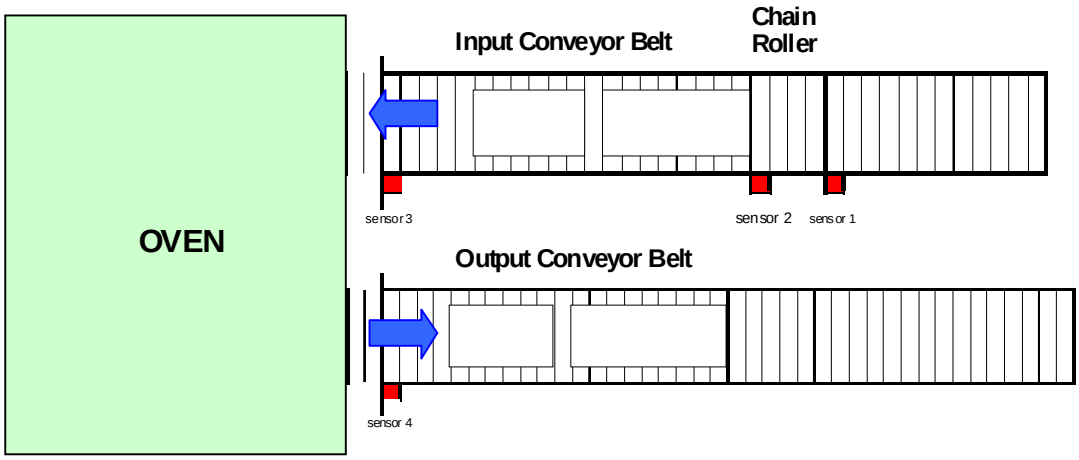

Fig. 1 – Diagram of the plant that must be controlled.

2. METHODOLOGY

The work is based on the ability of evolutionary systems to obtain optimal structures through the use of imprecise fitness functions. These functions need not provide an exact measure of the optimality of the system, but just a relative indication of how well it is performing with respect to the other systems in the population.

The basic idea in this work is to introduce as fitness evaluator in the evolutionary system a simulator of the whole plant that provides a simulation of the operation of combinations of individuals under different load conditions and provides a fitness value at its output that is related to the high level system constraints we mentioned before (productivity, energy cost, etc…). This approach, as anyone who has used plant simulators knows may be very computer resource consuming, but as we are using standard simulators and the brunt of the processing is taken up by the simulation stage we can distribute the simulation tasks in a cluster of computers and thus achieve reasonable execution times.

In this particular case we will be evolving digital electronic circuits made up of logic gates and flip flops so that temporal aspects of the sensor signals can be taken into account. The inputs to the circuits will be the data coming from the sensors positioned throughout the plant or production line to be controlled. The outputs to the circuits will act on certain control points modulating the activities of different machines, conveyor belts, and chain rollers in the line.

The evolutionary algorithms operate over variable length chromosomes and the encoding they contemplate is not direct, as is traditional in genetic algorithms, but rather, we encode the construction of binary trees. Thus, we provide certain discrimination between the genotype and phenotype. All the circuits will correspond to a tree per output where the root is the output and the nodes as we branch out correspond to the different digital elements (nand gates, type D flip flops, etc.) participating in the structure of the circuit. Obviously, there will be one tree per output, that is, there will be one tree per element to be controlled or modulated. There is no restriction on the number of inputs a given circuit may use in order to achieve the appropriate values for its outputs. It can choose to use all the possible inputs, just one of them or any number it requires.

The genetic operators employed by the evolutionary system correspond to traditional operators adapted to the peculiarities of binary tree type phenotypes. These operators act over the phenotype. In the case of the crossover operator, it will take portions of two parent

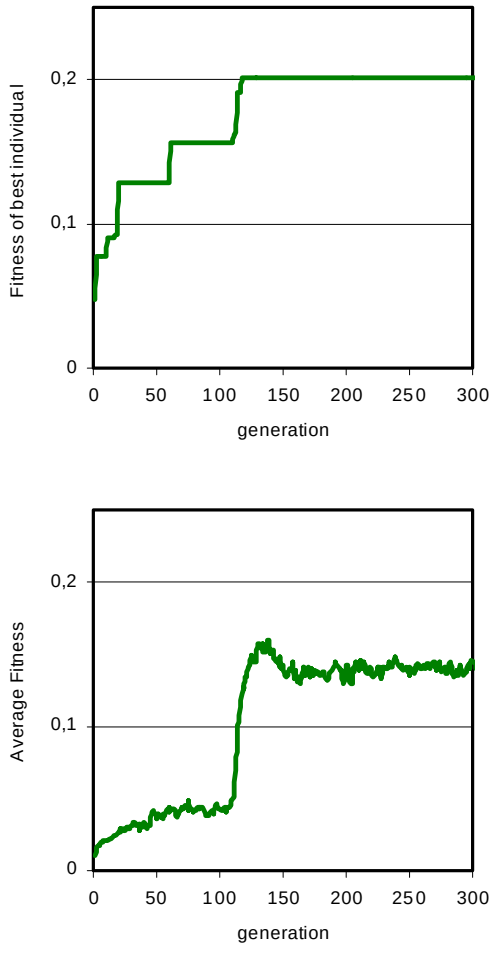

Fig. 2 – Maximum and average fitness of the population throughout the evolution process.

trees and join them together into an offspring. The mutation operators will change node values and add or delete subtrees or connections.

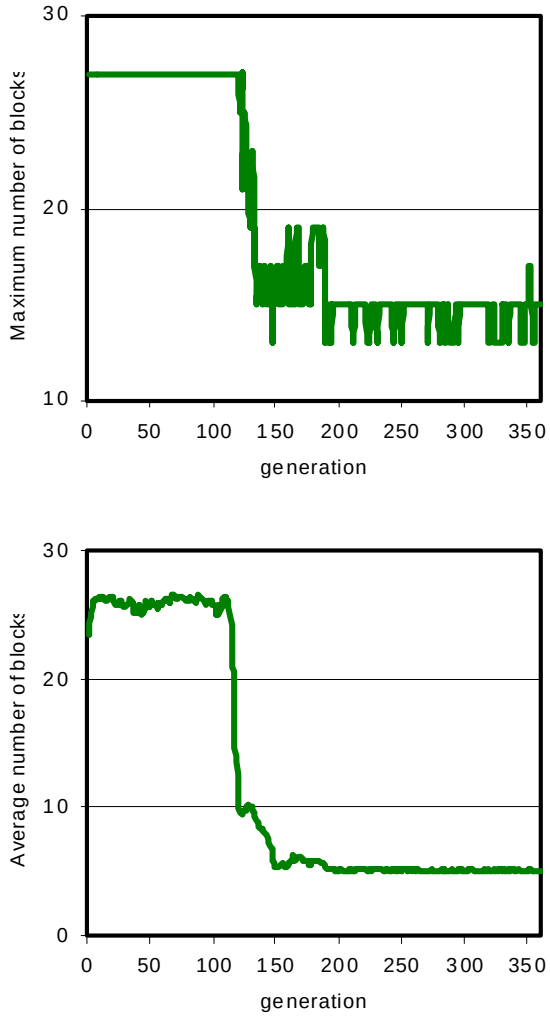

Fig. 3 – Maximum and average number of blocks that make up the individuals in the population throughout the evolution process.

In order to evaluate the resulting individuals using fitness measurements corresponding to the operation of the whole plant or line it is necessary to evaluate the set of controllers acting on the plant or a simulation of it for different conditions of workload and environment. To achieve this objective, we run simulations of the plants for which the circuits are being developed on *Extend* (production plant simulator) and provide different sets of workloads to the plant so that we can evaluate the global parameters to be taken as fitness in each case in a reliable manner.

This is obviously the most computationally intensive operation of the system. Each individual in the genetic population must be evaluated each generation of evolution. To make the system viable in normal situations the whole evolutionary tool was programmed in a distributed manner using MPI (message passing interface) libraries so that the evaluations could be distributed among a cluster of computers.

3. EXPERIMENT

In this particular problem we want to control a plant for a drying oven that takes boards of arbitrary size that have undergone a lacquering operation. As shown in Fig. 1 the ensemble is made up of two conveyor belts, a chain roller and an oven. The oven has a fixed load/unload sequence and can accept up to fifty sets of boards. There are four sensors, located as shown in Fig. 1 by the red squares and the objective is to obtain three electronic circuits that control the two conveyor belts and the chain roller and provide for the best global productivity without any pile-ups or unnecessary waits. This is, we want optimal productivity given the constraints of the oven with minimum energetic expense. It is important to note here that when the sequencing of the operations of the different elements is not perfect a pile up of boards may occur when the oven is not ready to accept inputs. This pile up is very costly as the plant must be stopped, the piled up boards taken out manually and thrown away. Consequently, this event must be avoided at all costs. In fact, this is the reason the company that runs the plant consulted us. They could find no way of avoiding pile ups without having to resort to very slow speeds and consequently lower productivities.

This plant will be controlled through the actuation over the speed of the two conveyor belts and the input chain roller. The three controllers will be three sequential digital circuits whose inputs are any or all of the four sensor values (chosen by the automatic procedure) and their outputs the speed actuation commands to their respective belts or roller. We allow the evolutionary system to make use of nand gates and type D flipflops. The sensors are infrared sensors that detect the presence or absence of a board beside them.

As commented in the methodology section, the GA accepts variable length chromosomes encoding the three circuits that are needed. The encoding corresponds to a binary tree representation, as each one of the nodes participating in the circuits (in this case we have constrained the possible elements to nand gates and type D flip flops) present two inputs and one output. The leaves of the tree are the sensors and the root node corresponds to the output of the circuit.

The fitness of each chromosome was obtained by creating the circuits (phenotype) it represents, introducing them in a model of the plant implemented in *Extend* (standard manufacturing plant simulator) and running a set of simulations with different sequences of boards.

The fitness function for each individual evaluates the weighed sum of three factors where we included:

- The energy used by the motors that control the oven input and output transports. The energy consumed is calculated as the average value of the difference between the speeds and their minimum value.
- The delays induced in the unload operation of the oven when the output transport is busy and in the load operation of the oven when the input transport is busy.
- The productivity of the line.

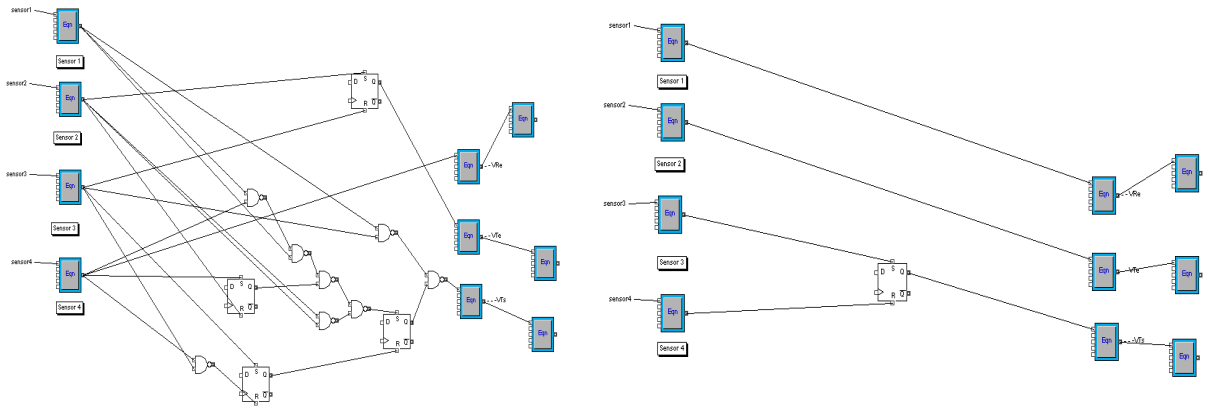

Fig. 4 – Left, circuit after generation 120. Right, circuit after generation 160.

Table 1. Some data on the performance of the resulting circuits

Generation	Individual	Producti-vity	Pile ups Unloading	Pile ups loading	Energy input belt	Energy input roller	Energy output belt	Nand gates	Flip-flops
111	300	111	0	0	0.815	2.13	3.49	8	4
360	300	111	0	0	4.5	1.13	5.4	1	0

The population employed was 300 individuals and we have considered 1% mutation probability. It was observed that evolution progressed much better if no restriction was initially placed on the size of the circuit until good solutions were obtained. From that point on smaller circuits with equal or better fitness were prioritized. In addition to using the fitness function to establish this priority, we introduced a bias in the mutation operator. In Fig. 2 we display the evolution of fitness for the best individual and the average fitness of the population during the automatic design process. Fig. 3 depicts the evolution of the number of elements in the circuits in the population for the best individual and the average of the population.

Initially the circuits grow in size while improving in performance. When performance is good, around generation 120, a new fitness and mutation term, which produces a bias towards smaller circuits, is introduced. This allows the circuits to become smaller if they preserve fitness. It is a strategy that avoids the problems that adding this constraint to the fitness function alone would introduce in terms of prioritizing smaller circuits that did not produce such good results in the operation of the plant. This introduction of the mutation term can be observed in the average fitness graph, around generation 130. It tends to reduce average fitness as more of the population are mutated towards smaller circuits in search for the good ones. Obviously, as we are using an elitist GA, the best fitness is preserved.

The resulting circuits are quite complex in generation 111 (see Fig. 4 left for a diagram of the circuit and Table 1 for its characteristics) but become really simple after the introduction of the bias without any loss of fitness. In fact,

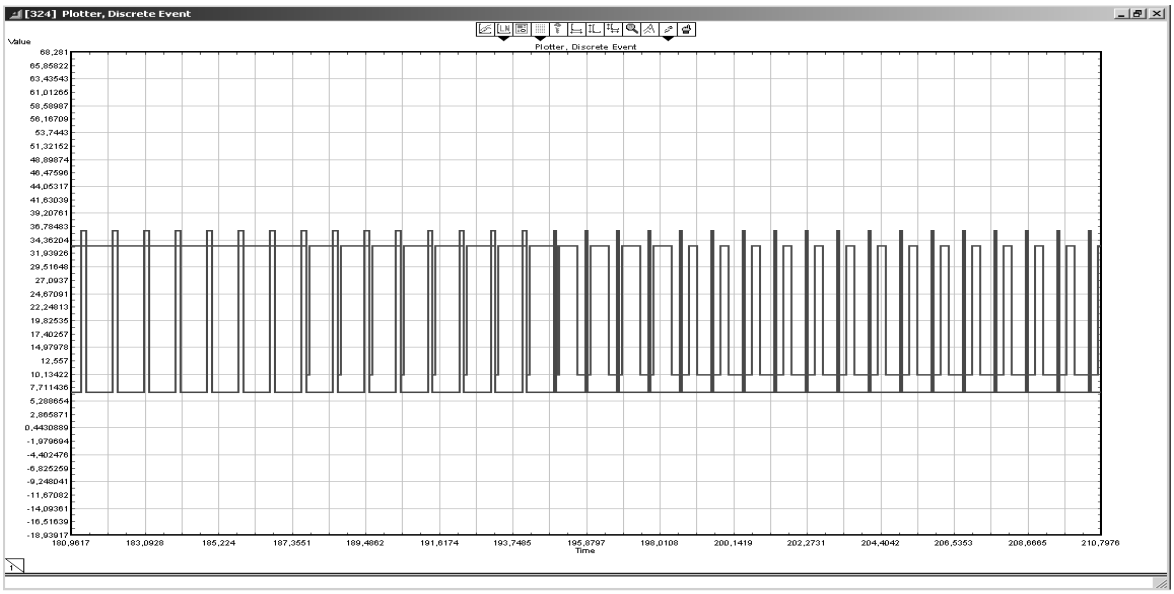

Fig. 5 – control signals provided by the resulting circuits for the input and output conveyor belts.

the resulting plant operates with the maximum theoretical productivity and, what is more important, there are never any pile ups no matter what the sequence of boards was, which was the main problem before these new circuits were introduced (see Fig. 4 right for the final circuit and Fig. 5 for an example of a diagram of the signals it produces to control input and output belts). The resulting circuits were compared to a set of controllers obtained by a group of engineers after exhaustive study and simulation of the plant for the same task and they perform either equally or better depending on the set. The main point here is that the circuits that were automatically obtained were produced after about 24 hours on a single PC (which means about one hour using a cluster of 30), whereas the manually obtained ones implied a process that took over three weeks.

4. CONCLUSIONS

This paper deals with obtaining the circuits for controlling manufacturing plants or lines in an automatic manner. The set of circuits must be obtained so that performance of the plant as a whole is maximized without having to explicitly define an individual training set or function for each individual circuit. To this end we have developed an evolutionary tool that integrates a standard plant controller and which, working over a binary tree representation of the circuits, can obtain in a simple and straightforward manner the set of controllers for the whole plant. In addition, the tool was programmed so as to be executable in a distributed fashion so that the computation times could be manageable in the case of complex systems. The results, tested on the case of a real line in a real manufacturing plant are very satisfactory as the system was able to obtain maximum productivity under any workload with very simple circuits.

5. ACKNOWLEDGEMENTS

This work was partially funded by This work was funded by Xunta de Galicia under project PGIDIT02PXIB10501PR the MCYT of Spain under projects TIC2000-0739C0404 and NATO under PST.CLG.978744.

6. REFERENCES

[1] J. Torresen. Evolvable Hardware as a new computer architecture *International Conference on Advances in Infrastructure for Electronic Business, Education, Science, and Medicine on the Internet (SSGRR 2002W)*. L`Aquila, Italy, January 2002.

[2] I. Harvey, P. Husbands, D. Cliff. Issues in Evolutionary Robotics, J-A. Meyer, H. Roitblat, and S. Wilson (Eds.), From Animals to Animats 2. *Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB 92)*, MIT Press, Cambridge, MA 1993, pp. 364-373.

[3] D.T. Cliff, P. Husbandsand, I. Harvey. Eds.: J-A. Meyer, H. Roitblat, S. Wilson. Evolving Visually Guided Robots. From Animals to Animats 2. *Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAP 92)*, MIT Press Bradford Books, Cambridge, MA 1993, pp. 374-383.

[4] R.D. Beer, J.C. Gallagher. Evolving Dynamical Neural Networks for Adaptive Behavior, *Adaptive Behavior*, Vol. 1, No. 1 (1992). pp. 91-122.

[5] T. Higuchi. Evolvable hardware: A first step towards building a Darwin machine. *In Proc. of the 2nd Int. Conf. on Simulated Behaviour*. MIT Press 1993, pp. 417-424.

[6] J. F. Miller. Digital filter design at gate-level using evolutionary algorithms. *In Proc. of the Genetic and Evolutionary Computation Conference*, 1999.

[7] M. Murakawa et al. The GRD chip: Genetic reconfiguration of DSPs for neural network processing, *IEEE Transactions on Computers*, Vol. 48, No. 6 (June 1999). pp. 628-638.

[8] J.D. Lohn, S.P. Colombano. A circuit representation technique for automated circuit design, *IEEE Trans. On Evolutionary Computation*, Vol. 3, No. 3 (September 1999), pp. 205-219.

[9] J. Torresen. Two-step incremental evolution of a digital logic gate based prosthetic hand controller. *In Evolvable Systems: From Biology to Hardware. Fourth Int. Conf., ICES'01*, Springer-Verlag 2001, Lecture Notes in Computer Science, Vol. 2210.